

CRYPT-01

crypt() is cryptographically weak; use stronger alternatives

Sean Barnum, Cigital, Inc. [vita¹]

Copyright © 2007 Cigital, Inc.

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5354 bytes

Attack Category	<ul style="list-style-type: none">• Encryption Assault	
Vulnerability Category	<ul style="list-style-type: none">• Cryptography	
Software Context	<ul style="list-style-type: none">• Cryptography	
Location		
Description	<p>crypt() is cryptographically weak and stronger alternatives should be preferred for the hashing of passwords.</p> <p>The standard crypt function can be used to hash passwords using a DES-based algorithm. It uses a password (up to 8 characters) and a salt (2 bytes) to generate a key to encrypt a string of zeroes using a slight variation of the DES algorithm. The problem is that the password can only be 8 characters long and there are 95 choices for each character. If a constant salt is used (which is the case if a salt is not explicitly provided to the crypt function), then an attacker can build a "dictionary" of all 2^{53} possible passwords and their hashes. Determining passwords then becomes a trivial table lookup. Even if the salt is different for each password, the dictionary size will increase to 2^{66}. In a few years, it should be practical to create a dictionary of that size. Also note that the effective dictionary size may be much smaller, since humans tend to use passwords that are easy to remember. Only considering passwords composed of English words would be a good way to obtain passwords from a database. Hence, the crypt function should generally not be used to hash passwords.</p>	
APIs	Function Name	Comments
	crypt	
Method of Attack		
Exception Criteria		

1. <http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html> (Barnum, Sean)

Solutions	Solution Applicability	Solution Description	Solution Efficacy
	When secure hashing (e.g., of passwords) is required	<p>Replace all calls to crypt() with a more secure version. The following libraries provide good choices:</p> <ul style="list-style-type: none"> * Cryptlib (Peter Gutmann) - not free, very robust * OpenSSL - freely available, popular choice * Crypto++ - open source * BSAFE - RSA Security's widely deployed commercial library <p>Some of the most widely available hash algorithms, though better than crypt(), may not be entirely secure. The MD5 algorithm has been discovered to have significant weaknesses, and the first hints of vulnerability have been found in SHA-1. Stronger algorithms in the SHA family are likely the most secure options at</p>	Effective, particularly if SHA-2 variant is used.

	present, but are not supported by all crypto libraries.				
Signature Details	char *crypt(const char *key, const char *salt)				
Examples of Incorrect Code	<pre>[...] hash = crypt(key, salt); [...]</pre>				
Examples of Corrected Code	<pre>/* SHA-1 algorithm from OpenSSL library. This is the best hash algorithm currently supported by OpenSSL, but SHA-2 should be preferred for high security applications. */ [...] unsigned char[SHA_DIGEST_LENGTH] hash; SHA1(passwordPlusSaltText, strlen(passwordPlusSaltText), hash) [...] /* Be aware that hash is a byte array rather than a null- terminated string as is returned by crypt(). If a printable string is needed, further encoding would be required, e.g., with base64 encoding. */</pre>				
Source Reference	<ul style="list-style-type: none"> Viega, John & McGraw, Gary. <i>Building Secure Software: How to Avoid Security Problems the Right Way</i>. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch. 11. 				
Recommended Resources	<ul style="list-style-type: none"> Wikipedia entry on MD5 algorithm² Wikipedia entry on SHA-1 algorithm³ 				
Discriminant Set	<table> <tr> <td>Operating Systems</td><td> <ul style="list-style-type: none"> UNIX (All) Windows </td></tr> <tr> <td>Languages</td><td> <ul style="list-style-type: none"> C C++ </td></tr> </table>	Operating Systems	<ul style="list-style-type: none"> UNIX (All) Windows 	Languages	<ul style="list-style-type: none"> C C++
Operating Systems	<ul style="list-style-type: none"> UNIX (All) Windows 				
Languages	<ul style="list-style-type: none"> C C++ 				

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@cigital.com¹.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. <mailto:copyright@cigital.com>